CLAIMS

What is claimed is:

1. A microprocessor, comprising:

a microcode unit for outputting control signals, for each of a plurality of instructions, required by said microprocessor for executing said instructions, the microcode unit comprising:

an instruction address input for receiving an instruction address;

a control variable input for receiving a control variable corresponding to a current state of the microprocessor;

a control signal input for receiving all the control signals output by the microcode unit for an immediately preceding instruction; and

a plurality of embedded logic circuits each dedicated for evaluating one unique type of instruction received by the microcode unit.

2. The microprocessor according to claim 1, wherein each of the embedded logic circuits includes:

a table for performing a table lookup in response to a received instruction; and

a controller responsive to the control variable, the control signals for an immediately preceding instruction, and to the table lookup for controllably setting each of the control signals required by the microprocessor for executing said received instruction.

3. The microprocessor of claim 2, wherein the controller includes:

means for setting a control signal to a "1" regardless of its immediately preceding value;

means for setting a control signal to a "0" regardless of its immediately

preceding value; and

means for not modifying a control signal from its immediately preceding value.

4. The microprocessor of claim 3, wherein the controller further includes:

means for setting a control signal to a data state.

5. A method of updating a design of a semiconductor chip at a hardware design language level (HDL) of simulation, to maximize an amount of logic that can be set to a previous cycle state, comprising:

automatically reading and setting a value of control signals on a per-cycle basis in a template and updating the HDL design with new data;

changing a first predetermined value of the template to be set with the previous state of the control signal; and

executing a test sweep to determine a "don't care" state of the control signals.

6. The method according to claim 1, further determining which of the control signals are not to be modified for each instruction.

7. The method according to claim 5, wherein said first predetermined value comprises a first non-zero value.

8. The method according to claim 5, wherein said "don't care" state indicates a state at which a respective control signal of said control signals maintains a value from its previous cycle.

9. A microcode unit in a microprocessor, for outputting control signals, for each of a plurality of instructions, required by said microprocessor for executing said instructions, the microcode unit comprising:

an instruction address input for receiving an instruction address;

a control variable input for receiving a control variable corresponding to a current state of the microprocessor;

a control signal input for receiving all the control signals output by the microcode unit for an immediately preceding instruction; and

a plurality of embedded logic circuits each dedicated for evaluating one unique type of instruction received by the microcode unit.

10. A microprocessor for a portable computer, for reducing power consumption, comprising:

a plurality of microcode units, said microcode units for outputting control signals, for each of a plurality of instructions, required by said microprocessor for executing said instructions;

means for maintaining selected control signals to values the same as in an immediately previous instruction cycle; and

means for passing a control variable to one of the microcode units.

11. The microprocessor of claim 10, further comprising embedded logic in the microcode units for testing for conditions of a respective opcode of an instruction.

12. A microprocessor system for reducing power consumption, comprising:

a plurality of microcode units each for receiving an instruction, each of said microcode units decoding said instruction to determine an instruction type, setting a value of each control signal within the microcode unit associated with the instruction,

wherein values set are one of "0", "1", or maintaining a previous value, and

wherein only control signals required for a function of an opcode of the instruction are changed, such that each opcode is a function of an

immediately previous opcode, wherein the control signals for each unit are changed only when their function interferes with the current opcode being executed, wherein embedded logic is included within the microcode and represents conditional checks for a particular opcode, where depending upon the value being set, the value is settable to different values; and

a set of logic trees for recombining the control signals and performing a logic operation on the control signals to produce a single final control signal.

13. The microprocessor according to claim 12, wherein the control signals are passed to the microcode unit are re-combined to form a single set of control signals, each of the microcode units have their own set of control signals.

14. The microprocessor according to claim 12, further comprising a latch for latching a single final control signal.

15. A method of reducing power in a portable microprocessor, including:

obtaining an opcode group and obtaining a control signal therefrom;

determining whether said control signal is active;

if said control signal is determined to be active, setting the control signal to active;

running a test case for the opcode group to determine whether the opcode passes; and

if said opcode group passes, marking the control signal.

16. The method according to claim 15, further including:

determining whether any other control signals exist for the opcode group;

when no more control signal exists, obtaining a next opcode group and testing and determining whether control signals of said next opcode should be

marked; and

when no more opcode groups exist, setting all marked control signals to active and executing a regression analysis thereon.

17. The method according to claim 16, further comprising:

when said regression analysis fails, performing debugging;

when said regression analysis passes, setting all marked signals to their previous states and executing another regression analysis thereon;

when said another regression analysis fails, performing debugging; and

when said another regression analysis passes, performing checking in the unit.

18. A method of creating a state machine control logic for a microprocessor, comprising:

providing a template for every function type within the microprocessor, said template being updated as s design specification on a per-cycle basis;

at a first function boundary, reading a current functional template from a hardware design language level;

obtaining a control signal from an opcode group, and determining whether the control signal is active;

if the control signal is not active, obtaining a next control signal from the opcode group, and if the control signal is active, selecting a first non-zero value of the template and changing the first non-zero value to be set to the previous state of the control signal;

searching a data base to locate all tests that have a function of said control signal and creating a test sweep;

updating by a new template, a temporary copy of the updated microcode function block;

running the test sweep with the previous state set for the one control

signal; and

if the test sweep passes, marking the control bit as a potential "don't care" value.

19. The method according to claim 18, further comprising:

examining and processing each control signal, setting each one individually, and running a test sweep automatically, to maintain the control signals found to be "don't care" and not setting function-critical signals;

after the template is fully swept on a per-bit basis and when all opcode groups are examined, setting all valid "don't care" bits;

re-running the test sweep to ensure that there is no bit-to-bit interconnections;

incrementing the function count , and creating a new template; finding tests, and completing analysis;

upon the function count reaching a maximum value, every function type being updated with the previous control signal;

executing all templates on the microcodes together with each other.;

executing a full regression to ensure that the opcode function still functions with the automatically-generated "don't care" verilog, and performing debugging if the regression fails;

if the full regression passes, then designating the new verilog as an official release design, and printing the templates in a readable format so that a complete documentation exists on a function-by-function basis of true "0s", true "1s", and previous "don't care" function on a bit-by-bit basis.

20. A method of providing a state machine decoding, comprising:

decoding a current opcode to provide a decode;

setting required functions signals;

setting exclusive functions outside of the current opcode to a previous state; and

latching results of the decode.

21. A signal-bearing medium tangibly embodying a program of machine-readable instructions executed by an apparatus to perform a method of updating a design of a semiconductor chip at a hardware design language level (HDL) of simulation, to maximize an amount of logic that can be set to a previous cycle state, said method comprising:

automatically reading and setting a value of control signals on a per-cycle basis in a template and updating the HDL design with new data;

changing a first predetermined value of the template to be set with the previous state of the control signal; and

executing a test sweep to determine a "don't care" state of the control signals.

22. A signal-bearing medium tangibly embodying a program of machine-readable instructions executed by an apparatus to perform a method of reducing power in a portable microprocessor, said method including:

obtaining an opcode group and obtaining a control signal therefrom;

determining whether said control signal is active;

if said control signal is determined to be active, setting the control signal to active;

running a test case for the opcode group to determine whether the opcode passes; and

if said opcode group passes, marking the control signal.